8 Fallunterscheidung

Lernziele In diesem Abschnitt lernst du:

- ▷ Mit dem Computer zwischen verschiedenen Fällen zu unterscheiden und dadurch auch Spezialfälle zu berücksichtigen.
- ▷ Programmcode nur in bestimmten Situationen ausführen zu lassen.

Einführung Ein Programm soll nicht immer alle Befehle der Reihe nach durcharbeiten, sondern manchmal eine Auswahl treffen und gewisse Befehle nur ausführen, wenn auch die Voraussetzungen dafür gegeben sind. In anderen Worten: Das Programm muss verschiedene Fälle unterscheiden können und dabei auch Spezialfälle berücksichtigen.

Stell dir z. B. vor, dein Programm soll die Wurzel einer Zahl x ziehen. Das geht nur, wenn die Zahl x nicht negativ ist! Es hat also Sinn, vor dem Wurzelziehen den Wert von x mit `if` zu überprüfen:

```
if x >= 0:
    Wurzel ziehen
```

Mit `if` hast du also beim Programmieren die Möglichkeit, auf spezielle Situationen gezielt zu reagieren. Dazu braucht `if` immer eine Bedingung, um entscheiden zu können, ob diese Situation wirklich eintrifft.

Das Programm Woran erkennst du, ob eine Zahl eine Quadratzahl ist? Und wie programmierst du den Computer so, dass er Quadratzahlen erkennt? In diesem Programm hier haben wir folgende Idee verwendet: Wenn du die Wurzel einer Quadratzahl ziehst, dann sind die Nachkommastellen alle Null.

Der ganze Trick funktioniert aber nur, wenn die Zahl nicht-negativ ist. Von negativen Zahlen können wir keine Wurzeln ziehen und das Programm würde abstürzen.

```
1 from math import *
2 zahl = 74
3 if zahl >= 0:
4     wurzel = sqrt(zahl)
5     kommateil = wurzel % 1
6     if kommateil == 0.0:
7         print "Zahl ist eine Quadratzahl."
8     if kommateil != 0.0:
```

```
9           print "Zahl ist keine Quadratzahl."
10  if zahl < 0:
11      print "Negative Zahlen sind nie Quadrate."
```

Ändere den Anfangswert `zahl = 74` in Zeile 2 ab und probiere verschiedene Werte aus. Mache dich so soweit mit dem Programm vertraut, dass du die `if`-Struktur wirklich verstehst!

Die wichtigsten Punkte Die `if`-Struktur hat immer eine Bedingung und darunter Programmcode, der eingerückt ist. Dieser eingerückte Code wird nur dann ausgeführt, wenn die Bedingung bei `if` erfüllt ist. Ansonsten überspringt Python den eingerückten Code.

```
if Bedingung:
    Code, der nur ausgeführt wird,
    wenn die Bedingung wahr ist.
```

Warum braucht es bei Vergleichen ein doppeltes Gleichheitszeichen? Weil das einfache Gleichheitszeichen für Python immer *eine Zuweisung ist.* `x = 3` *heisst also* in jedem Fall, *die Variable* x *soll den Wert 3 haben.*
`if x = 3` *bedeutet für Python übersetzt: «Die Variable* x *hat jetzt den Wert 3 und falls ja, dann...». Das hat nicht wirklich Sinn!*

Die Bedingung ist meistens ein Vergleich. Dabei ist speziell, dass du zwei Gleichheitszeichen brauchst, um zu prüfen, ob zwei Werte gleich sind!

`x == y`	gleich	`x != y`	ungleich
`x < y`	kleiner als	`x >= y`	grösser oder gleich
`x > y`	grösser als	`x <= y`	kleiner oder gleich

AUFGABEN

34. Schreibe ein Programm, das überprüft, ob eine Zahl gerade ist und entsprechend «gerade» oder «ungerade» auf den Bildschirm schreibt.

35. Schreibe ein Programm, das überprüft, ob ein gegebenes Jahr ein Schaltjahr ist. Achte darauf, dass dein Programm auch mit vollen Jahrhunderten (1600, 1700, etc.) richtig umgehen kann. Mit der Einführung des gregorianischen Kalenders 1582 wurde die Schaltjahrregelung nämlich so ergänzt, dass von den vollen Jahrhunderten nur diejenigen Schaltjahre sind, deren erste zwei Ziffern durch 4 teilbar sind.

36. Schreibe ein Programm, das zu einer Zahl alle Teiler sucht und ausgibt. Verwende dazu eine Schleife. In dieser Schleife prüfst du mit der «Division mit Rest» alle möglichen Teiler durch und schreibst diese möglichen Teiler auf den Bildschirm, wenn der Rest Null ist.